Indexed Compression

Arun Jose
joseverres42@gmail.com

Abstract

This paper examines a lossless text compression algorithm that works on the principle that in any text file with recognizable words, the vast majority of character sequences would be recorded in a comprehensive dictionary, and can thus be mapped to bits directly, taking advantage of information entropy.
The algorithm is tested on a sample book file, and compression factor and time are reported.

Contents

1 Introduction

Indexed compression is a form of lossless text compression that maps words in a text file to bit-numerical indexes in a comprehensive dictionary of the language (inclusive of slang, word forms, and contractions), as well as stores information about words unique to the file with additional indexes.

1.1 Principle

This algorithm is targeted at documents whose content is primarily recognizable words in a language, being relatively sparse in an information entropy sense.

This enables us to heavily optimize on the efficiency of knowledge encoding in a bit, leveraging high-level language patterns already encoded into the mind, and mapping more directly to it. Instead of relating bits to exhaustive combinations of alphabets, a comprehensive dictionary is chosen.

1.2 Word Archive

For the purposes of demonstrating this algorithm, a dictionary consisting of 41,284 words was chosen from Wiktionary's frequency lists [1].

The Oxford English Dictionary (Second Edition) contains 171,476 words [2], but due to reasons of free availability and necessity of frequency analysis, this shorter form was chosen to prototype the algorithm's implementation. There are **much** more comprehensive and more accurate frequency analysis, for dictionaries in the size of millions of words [3] - but which are expensive - so there is still scope for improvement on the efficiency of this factor, although it would be slightly offset the compression time.

2 Generating Dictionary

The words in the chosen dictionary are indexed by frequency rankings. In order to account for punctuations and numbers, and their frequencies (for which comparative analysis to the frequencies of words was not found, and was therefore approximated), they are divided into three groups: Group 1, which contains punctuations more frequently used than most common words; Group 2, which contains digits, and further punctuations; and Group 3, which stores further numbers in part of the remaining bit space to cut down on storage expense of storing larger numbers.

The dictionary containing the words and the first two groups extends up to 16-bit indexes, with over 24,000 indexes to spare in that range. As space has to be allowed for words unique to the text, numbers up to 24,107 are stored, leaving 128 empty bit-spaces for new words.

The Python3 code for generating this dictionary from a list of these words can be found here.

2.1 Sources

The words and frequencies were sourced from here.

2.2 Dictionaries

The dictionary mapping words to their bit-number indexes for compression can be found here.

The dictionary mapping bit-number indexes to their corresponding words for decompression can be found here.

3 Compression Factor and Time

This algorithm was tested on a plaintext format of *Pride and Prejudice* by Jane Austen, obtained from Project Gutenberg [4]. Owing to the smaller size of the dictionary, there were 1,152 unique words - archaic usage, word forms, etc.; the majority of which are covered by more comprehensive dictionaries.

The text file is 685,570 bytes in length. After compression, it was reduced to 130,374.

This implies a compression factor of 5.258487, or approximately 81% reduction in size.

The average compression time per byte is 57075.97473 nanoseconds.

4 Sample

The implementation of this algorithm in Python3 as well as the text file can be found in the following repository.

Repository

Book

Algorithm

5 Conclusion

If we define the quality of a compression method by four characteristics - whether it is lossless, the compression factor, compression time, and range of applicability, then this compression technique works well on the first three, with a relatively high compression factor and low compression time, and sacrifices on range, as it functions well only on text files with recognizable words, and not arbitrary sequences of characters.

References

[1] Wiktionary. “Frequency lists”.

[2] Lexico. “How many words are there in the English language?”

[3] Word Frequency. “Frequency data on the Corpus of Contemporary American English (COCA)”.

[4] The Project Gutenberg. “Pride and Prejudice”.